

Parsing

Phase	Input	output
lexer	string of characters	string of tokens
Parser	string of tokens	Parse tree

Parser Roles

- 1- يحدد (string of tokens) هل هي (Parse tree) أم لا.
- 2- يميز بين (Valid and non valid string of tokens).

(CFG) Context Free grammars

* Programming language constructs have recursive structure.

* Context-Free grammars are natural notation for describing this recursive structure.

CFG consist of

1) set of terminals (T)

→ الشكل الذي عليه يمثل اللغة بتابع.

2) set of Non-terminals. (NT)

3) A start symbol S (a non-terminal)

4) set of production

هـ وظيفتها: تعمل (replace) في ال (string) (الناتج).

$S \rightarrow Ab$

Terminals $\leftarrow b, c$

$A \rightarrow Bc$

Non terminals $\leftarrow S, A, B, C$

$B \rightarrow Cc$

$C \rightarrow \epsilon$

\hookrightarrow that string doesn't accept except
 b 's or c 's.

هـ وظيفتها ال Grammar

1- يعرف كل ال (expressions) التي تولد اللغة الخاصة به.

2- يعمل استبعاد أو استثناء لكل ال (expressions) التي مشعيا.

* How to Driven any Expression

(1) ابدأ ب (string) يحتوى الرمز " S " (start symbol)

(2) استبدل ال Non-terminal وضع ما ينفرد T or NT or ϵ

(3) كرر رقم (2) حتى تصل لحالة عدم وجود Non terminal في ال (string)

* مقدرش اعل (replacement) لای (terminal)
بواسطة ال (Production tool).

* كل اما بوجل (terminal) يكون ثابت معنا.

* if you wanna to make derivation for
if id then id else id fi then id else id fi

$Expr \rightarrow \text{if } \underline{Expr} \text{ then } Expr \text{ else } Expr \text{ fi}$

$\rightarrow \text{if } (\text{if then } Expr \text{ else } Expr \text{ fi})$

then Expr else Expr fi

* which of strings are in language of given CFG

1) abcba \Rightarrow xx

2) acca \Rightarrow xx

3) aba \Rightarrow ✓✓

4) abcbcb a \rightarrow ~~xx~~ xx

$S \rightarrow a X a$

$X \rightarrow \epsilon$

$| b Y$

$Y \rightarrow \epsilon$

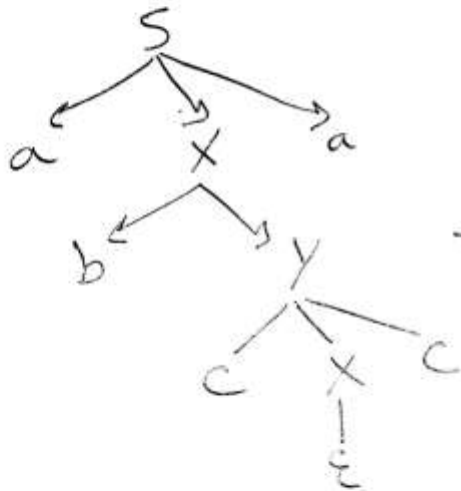
$| c X c$

3

من هنا خذهم نقطة نقطة

a) $abcba$

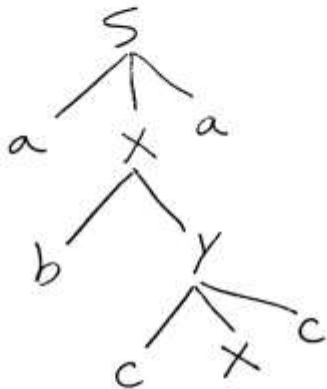
هناك دل نشوف ال (CFG) اللي عندنا عاليمين في الصفحة السابقة بتتحقق مع النقطة a ام لا .



لا تتألف $abccca$

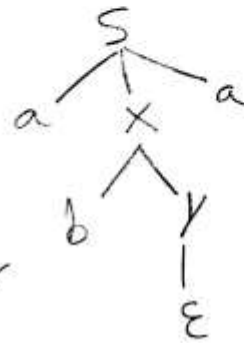
unaccepted

b) $acca$



لا تتحقق 'acca'

c) aba



$aba \Rightarrow$ accepted

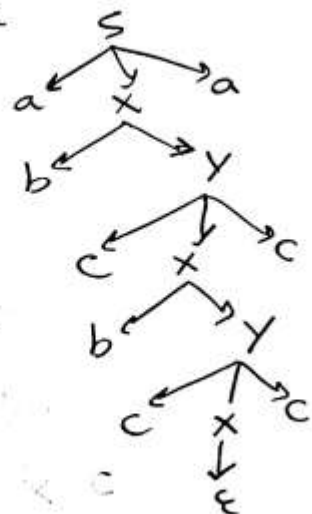
~~aba~~

d)

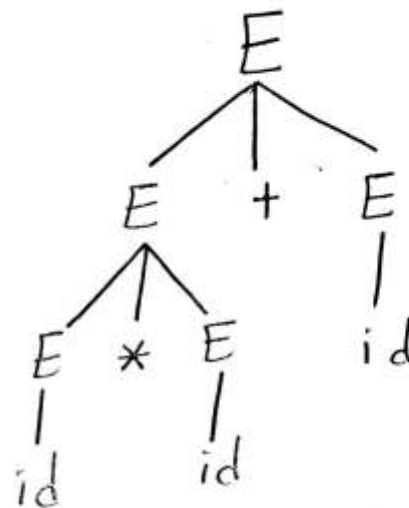
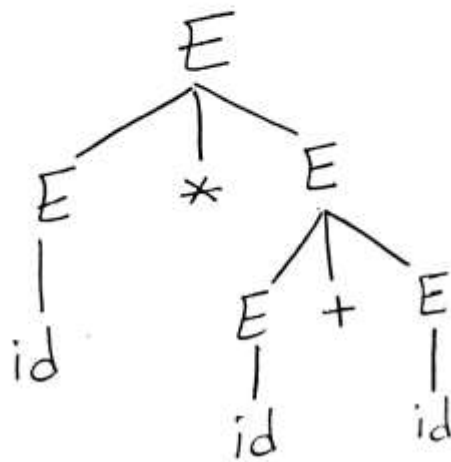
4

$abebccca$

not accepted

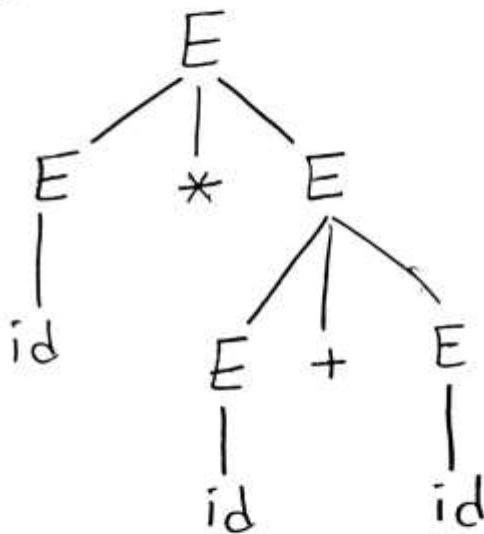


Ambiguous

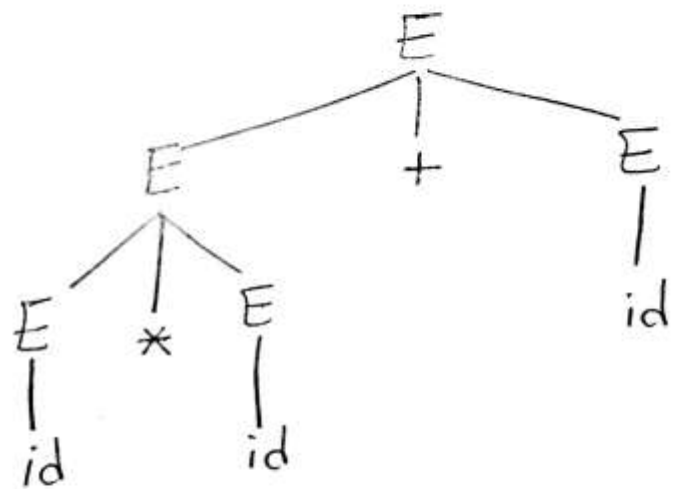


← معناه اني دومت بال (derivation) من النتيجة
 لكن ب (2-parsing trees) مختلفين .

Right most derivation



left most derivation



(non terminal) له تشتغل على ال [5]
 في ال (right) ، نفعله
 استبداله .

* أمثلة ال (Ambiguous) متواجدة في ال (Sheets).

* لا يوجد خطرات ثابتة في تمنع وجود ال (Ambiguous)

بس الحل الأقرب! نك تعمل (rewrite) لا (grammar) بحيث ~~ت~~ (Avoid Ambiguity).

EX

$S \rightarrow AaA \mid AbA$ ← (Ambiguous) دي

$A \rightarrow c \mid s$

الحل ← تستبدل $B \leftarrow AbA$ و $s \sim$

$S \rightarrow AaA \mid B$

~~$B \rightarrow AbA$~~ $B \rightarrow AbA$

$A \rightarrow c \mid s$

~~MIF~~ The dangling else

→ else matches the closest then.

$S \rightarrow MIF$

$\mid UIF$

(all then matched with an else)

(some then is unmatched)

$MIF \Rightarrow IF \ E \text{ then } \underbrace{MIF}_{\downarrow} \text{ else } \underbrace{MIF}_{\downarrow}$
if-then else if-then else

6

* if E then E else E

→ if E then UIF else MIF

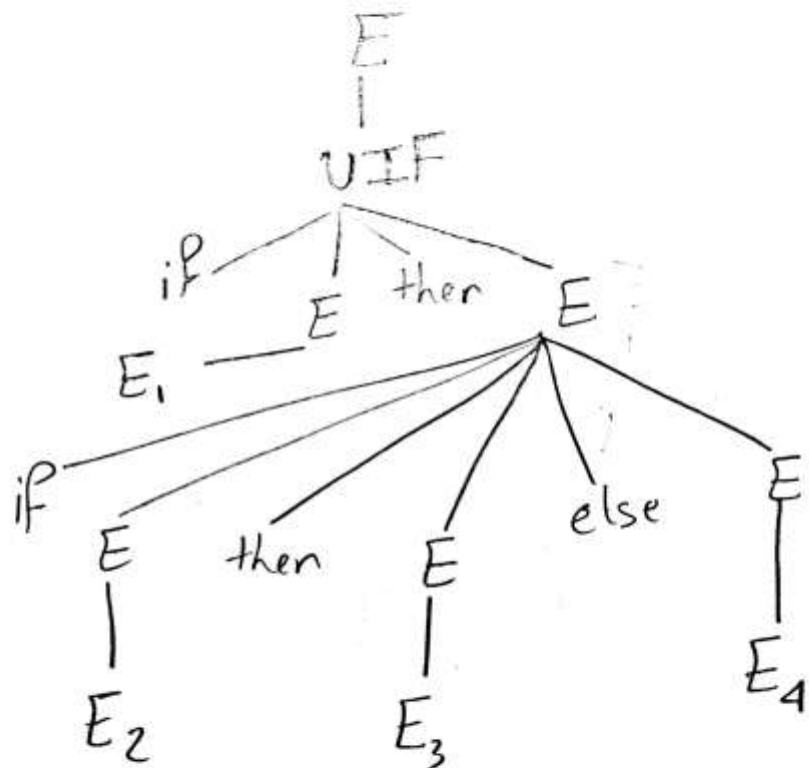
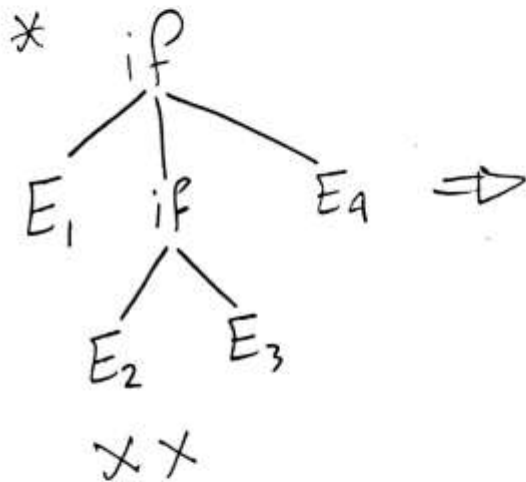
if then

لأنه (else) تبع الـ (if) ← غير مقبول

→ if E then MIF else UIF

if then else

accepted



منه أحد وظائف الـ (Compiler) إنه يعمل (handling)
لا (non-valid program or errors)

Panic mode [1]

من يقدم يعمل تتجاهل لا (tokens) حتى يصل لنقطة
واضحة تجعل الجملة صحيحة.

[Ex]

$$(1 + + 2) + 3$$

منه المثال لليمين اثم ثم تتجاهل + القادمة
حتى تصل لـ 2 ويكمل الجملة ويعطى رسالة إن فيه
(syntax error) في مكان معين.

$$1 + \xrightarrow{\text{حتى تصل}} 2$$

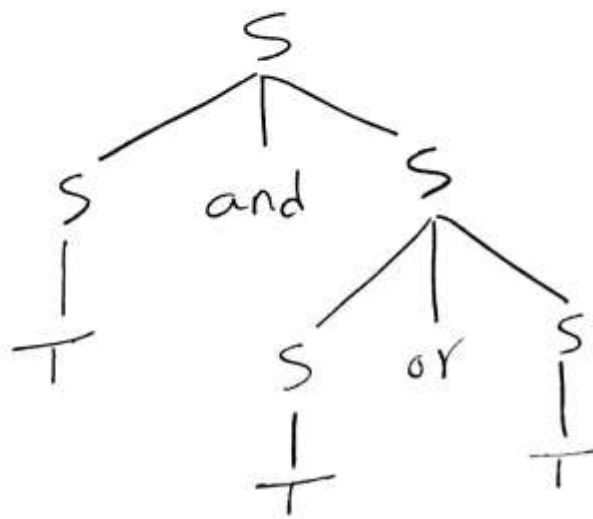
من يتبقى (error production)
→ on Lecture 5 (slides) last

3 - pages.

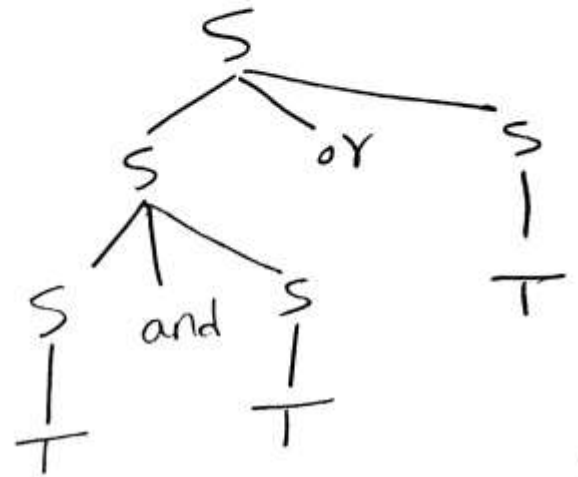
[8]

P. * Precedence

← ! يه اللي إنتفض الأول



(or) لها أولوية أعلى.

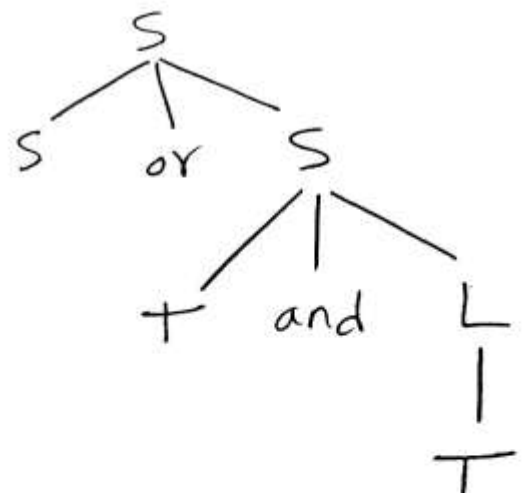


له (and) لها أولوية أعلى.

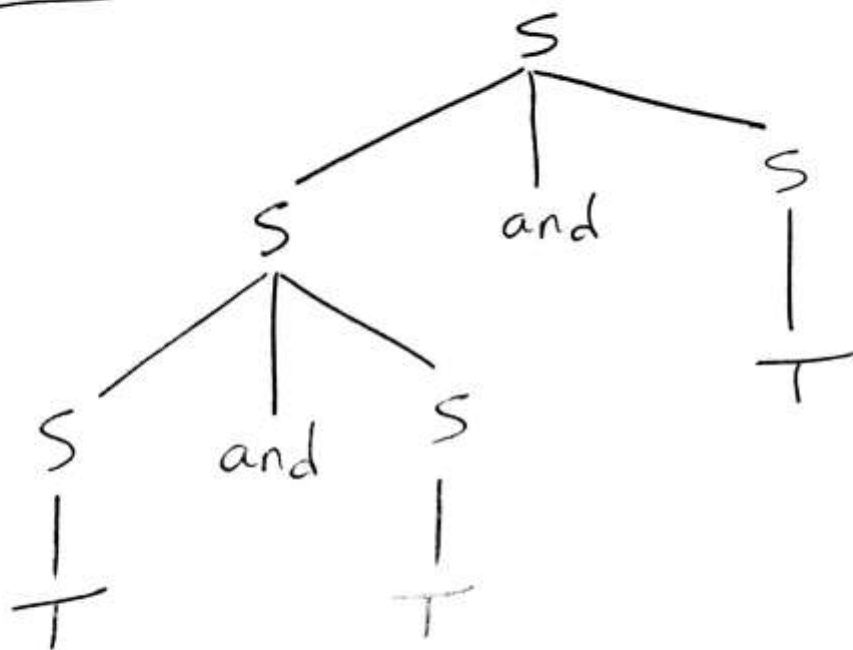
← لو عايز أجبره إنه يخلي (and) لها أولوية
عند ال (or)

$$S \rightarrow S \text{ or } S \mid L$$

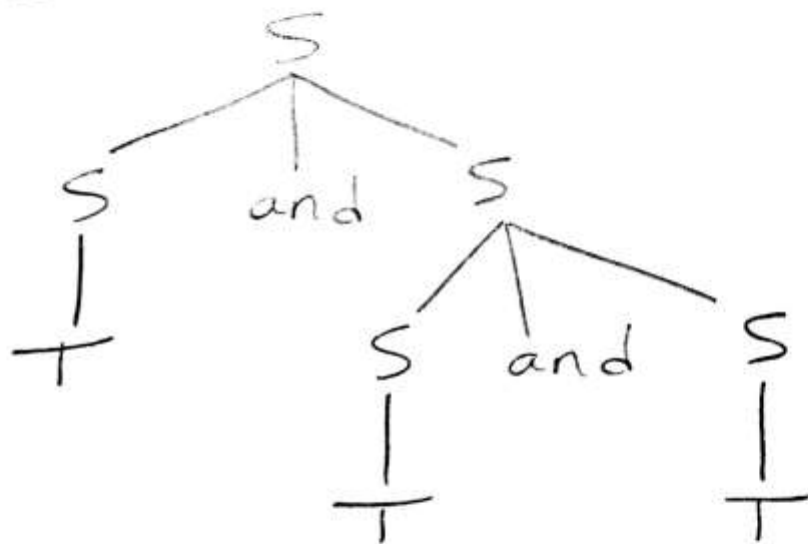
$$L \rightarrow \text{true and } L \mid \text{true}$$



* Left Associative



* Right Associative



10